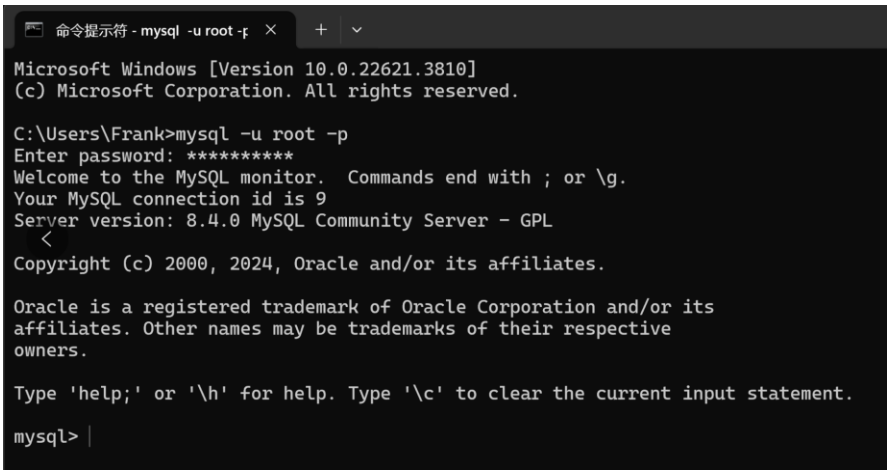
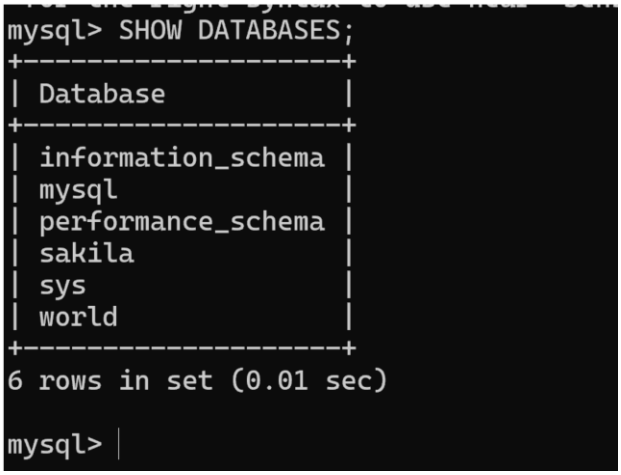
Open MySQL in cmd, **command: mysql -u root -p**



Next, look at all the current databases, **command: SHOW DATABASES;**



First, the important part is importing mysql.connector. This is crucial as it allows us to connect to the database. **The syntax mysql.connector.connect helps us establish this connection. Next, the cursor is used to read data from our Excel file and import it into two tables: one for Sensors and another for Measurements.** Then, each row is iterated over and imported from Excel into MySQL. Finally, we commit to the changes using commit and close the connection.

```
import pandas as pd
import mysql.connector
import numpy as np

# Connect to the MySQL database
connection = mysql.connector.connect(
    host='localhost',
    user='root',
```

```
    password='123Wjb456+',
    database='sensor_data'
)

cursor = connection.cursor() # Create a cursor object to execute SQL statements

file_path = 'C:/Users/Frank/Desktop/New_DHT11_Data_Collection.xlsx' # Read data
from the Excel file
sensors = pd.read_excel(file_path, sheet_name='Sensors') # Read the 'Sensors'
sheet
measurements = pd.read_excel(file_path, sheet_name='Measurements') # Read the
'Measurements' sheet

# Remove duplicate primary key records
sensors = sensors.drop_duplicates(subset=['Sensor Key'])
measurements = measurements.drop_duplicates(subset=['ID'])

try:
    for index, row in sensors.iterrows(): # Iterate through each row of the
Sensors DataFrame and insert data into the Sensors table, ignoring duplicate
primary keys
        sql = "INSERT IGNORE INTO Sensors (Sensor_ID, Location, Sensor_Key)
VALUES (%s, %s, %s)" # Define the insert statement, using INSERT IGNORE to avoid
duplicate primary key errors
        params = (row['Sensor ID'], row['Location'], row['Sensor Key']) # Extract
data from the current row
        print(f"Executing SQL: {sql} with params {params}") # Print the SQL
statement and parameters to be executed for debugging purposes
        cursor.execute(sql, params) # Execute the insert operation

    for index, row in measurements.iterrows(): # Insert data into the
Measurements table, ignoring duplicate primary keys
        sql = """INSERT IGNORE INTO Measurements (ID, Temperature, Humidity,
Timestamp, TimeZone, Temperature_Change_Rate,
                  Humidity_Change_Rate, Comfort_Index, Location, Sensor_ID,
Comfort_Level)
                  VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)""" #
Define the insert statement, using INSERT IGNORE to avoid duplicate primary key
errors

        params = (row['ID'], row['Temperature'], row['Humidity'],
row['Timestamp'], row['TimeZone'],
                  row['Temperature Change Rate'], row['Humidity Change Rate'],
row['Comfort Index'], row['Location'], row['Sensor ID'], row['Comfort Level']) #
Extract data from the current row
```

```
        print(f"Executing SQL: {sql} with params {params}") # Print the SQL statement and parameters to be executed for debugging purposes
        cursor.execute(sql, params) # Execute the insert operation

    connection.commit() # Commit changes to the database to ensure all insert operations are saved

except mysql.connector.Error as err: # Catch and print database operation errors
    print("Error: {}".format(err))

finally:
    cursor.close() # Close the cursor
    connection.close() # Close the connection
```

```
mysql> CREATE DATABASE sensor_data;
Query OK, 1 row affected (0.02 sec)

mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sakila             |
| sensor_data        |
| sys                |
| world              |
+--------------------+
7 rows in set (0.00 sec)

mysql> USE sensor_data;
Database changed
mysql> CREATE TABLE Sensors (
    ->  Sensor_Key INT PRIMARY KEY,
    ->  Sensor_ID VARCHAR(255) NOT NULL,
    ->  Location VARCHAR(255) NOT NULL
    -> );
Query OK, 0 rows affected (0.04 sec)
```

```
mysql> CREATE TABLE Sensors (
    ->  Sensor_Key INT PRIMARY KEY,
    ->  Sensor_ID VARCHAR(255) NOT NULL,
    ->  Location VARCHAR(255) NOT NULL
    -> );
Query OK, 0 rows affected (0.04 sec)

mysql> CREATE TABLE Measurements (
    ->     ID INT PRIMARY KEY NOT NULL,
    ->     Temperature FLOAT NOT NULL,
    ->     Humidity INT NOT NULL,
    ->     Timestamp DATETIME NOT NULL,
    ->     TimeZone VARCHAR(255) NOT NULL,
    ->     Temperature_Change_Rate VARCHAR(255),
    ->     Humidity_Change_Rate VARCHAR(255),
    ->     Comfort_Index FLOAT,
    ->     Location VARCHAR(255) NOT NULL,
    ->     Sensor_ID INT NOT NULL,
    ->     Comfort_Level VARCHAR(255),
    ->     FOREIGN KEY (Sensor_ID) REFERENCES Sensors(Sensor_Key)
    -> );
Query OK, 0 rows affected (0.07 sec)

mysql>
```

Next, show the databases, **SHOW DATABASES sensor_data;**

Then enter sensor_data and show the tables (one is Sensors and the other is Measurements).

USE sensor_data; **CREATE TABLE Sensors** **CREATE TABLE Measurements**

DESCRIBE Sensors; **DESCRIBE Measurements;**

```
mysql> DESCRIBE Measurements;
+-------------------------+-------------+------+-----+---------+-------+
| Field                   | Type        | Null | Key | Default | Extra |
+-------------------------+-------------+------+-----+---------+-------+
| ID                      | int         | NO   | PRI | NULL    |       |
| Temperature             | float       | NO   |     | NULL    |       |
| Humidity                | int         | NO   |     | NULL    |       |
| Timestamp               | datetime    | NO   |     | NULL    |       |
| TimeZone                | varchar(255)| NO   |     | NULL    |       |
| Temperature_Change_Rate | varchar(255)| YES  |     | NULL    |       |
| Humidity_Change_Rate    | varchar(255)| YES  |     | NULL    |       |
| Comfort_Index           | float       | YES  |     | NULL    |       |
| Location                | varchar(255)| NO   |     | NULL    |       |
| Sensor_ID               | int         | NO   | MUL | NULL    |       |
| Comfort_Level           | varchar(255)| YES  |     | NULL    |       |
+-------------------------+-------------+------+-----+---------+-------+
11 rows in set (0.01 sec)
```

```
mysql> DESCRIBE Sensors;
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| Sensor_Key | int          | NO   | PRI | NULL    |       |
| Sensor_ID  | varchar(255) | NO   |     | NULL    |       |
| Location   | varchar(255) | NO   |     | NULL    |       |
+------------+--------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Show the database contents in the table

SELECT * FROM Sensors; **SELECT * FROM Measurements;**

```
mysql> SELECT * FROM Sensors;
+------------+-----------+-------------+
| Sensor_Key | Sensor_ID | Location    |
+------------+-----------+-------------+
|          1 | DHT11     | bedroom     |
|          2 | DHT22     | balcony     |
|          3 | DHT11     | First Floor |
|          4 | DHT22     | bathroom    |
|          5 | DHT11     | backyard    |
+------------+-----------+-------------+
5 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM Measurements;
+------+-------------+----------+---------------------+----------+-----------------------+----------------------+---------------+----------+-----------+---------+
| ID   | Temperature | Humidity | Timestamp           | TimeZone | Temperature_Change_Rate | Humidity_Change_Rate | Comfort_Index | Location | Sensor_ID | Comfort_Level |
+------+-------------+----------+---------------------+----------+-----------------------+----------------------+---------------+----------+-----------+---------+
|    1 |        28.5 |       56 | 2024-06-21 11:24:32 | morning  | no change             | no change            |       -2.0541 | bedroom  |         1 | High    |
|    2 |        28.5 |       56 | 2024-06-21 11:24:34 | morning  | no change             | no change            |       -2.0541 | bedroom  |         1 | High    |
|    3 |        28.5 |       56 | 2024-06-21 11:24:36 | morning  | no change             | no change            |       -2.0541 | bedroom  |         1 | High    |
|    4 |        28.5 |       56 | 2024-06-21 11:24:38 | morning  | no change             | no change            |       -2.0541 | bedroom  |         1 | High    |
|    5 |        28.5 |       56 | 2024-06-21 11:24:40 | morning  | no change             | no change            |       -2.0541 | bedroom  |         1 | High    |
|    6 |        28.5 |       56 | 2024-06-21 11:24:43 | morning  | no change             | no change            |       -2.0541 | bedroom  |         1 | High    |
|    7 |        28.5 |       56 | 2024-06-21 11:24:45 | morning  | no change             | no change            |       -2.0541 | bedroom  |         1 | High    |
|    8 |        28.5 |       56 | 2024-06-21 11:24:47 | morning  | no change             | no change            |       -2.0541 | bedroom  |         1 | High    |
|    9 |        28.5 |       56 | 2024-06-21 11:24:49 | morning  | no change             | no change            |       -2.0541 | bedroom  |         1 | High    |
|   10 |        28.5 |       56 | 2024-06-21 11:24:51 | morning  | no change             | no change            |       -2.0541 | bedroom  |         1 | High    |
|   11 |        28.5 |       56 | 2024-06-21 11:24:53 | morning  | no change             | no change            |       -2.0541 | bedroom  |         1 | High    |
|   12 |        28.5 |       56 | 2024-06-21 11:24:55 | morning  | no change             | no change            |       -2.0541 | bedroom  |         1 | High    |
|   13 |        28.5 |       56 | 2024-06-21 11:24:57 | morning  | no change             | no change            |       -2.0541 | bedroom  |         1 | High    |
| 1788 |        24.1 |       65 | 2024-06-22 18:19:44 | evening  | no change             | no change            |      -11.8441 | bedroom  |         1 | Medium  |
| 1789 |        24.1 |       65 | 2024-06-22 18:19:46 | evening  | no change             | no change            |      -11.8441 | bedroom  |         1 | Medium  |
| 1790 |        24.1 |       65 | 2024-06-22 18:19:48 | evening  | no change             | no change            |      -11.8441 | bedroom  |         1 | Medium  |
| 1791 |        24.1 |       65 | 2024-06-22 18:19:50 | evening  | no change             | no change            |      -11.8441 | bedroom  |         1 | Medium  |
| 1792 |        24.1 |       65 | 2024-06-22 18:19:52 | evening  | no change             | no change            |      -11.8441 | bedroom  |         1 | Medium  |
| 1793 |        24.1 |       65 | 2024-06-22 18:19:55 | evening  | no change             | no change            |      -11.8441 | bedroom  |         1 | Medium  |
| 1794 |        24.1 |       65 | 2024-06-22 18:19:57 | evening  | no change             | no change            |      -11.8441 | bedroom  |         1 | Medium  |
| 1795 |        24.1 |       65 | 2024-06-22 18:19:59 | evening  | no change             | no change            |      -11.8441 | bedroom  |         1 | Medium  |
| 1796 |        24.1 |       65 | 2024-06-22 18:20:01 | evening  | no change             | no change            |      -11.8441 | bedroom  |         1 | Medium  |
+------+-------------+----------+---------------------+----------+-----------------------+----------------------+---------------+----------+-----------+---------+
1796 rows in set (0.00 sec)
```

Pagination

One aspect to discuss is pagination. **During the visualization phase, our database content is extensive, and it is impossible to display all the data on a single page. Therefore, pagination will be used.** **Pagination, in simple terms, means dividing large amounts of data into smaller chunks for transmission, enhancing performance and user experience.**

In Django, this can be achieved using PageNumberPagination. As we can see, utilizing different functionalities is straightforward because most of them already exist in libraries. We just need to import and use the appropriate libraries. For instance, we have discussed pandas, numpy, scipy, mysql-connector from the beginning. Django itself is a library, along with the Django REST framework, and so on. Therefore, the key is to effectively use each library.

```python
from rest_framework.pagination import PageNumberPagination
from rest_framework import viewsets

# Custom pagination class, inheriting from PageNumberPagination
class Pagination(PageNumberPagination):
    page_size = 10 # Number of items displayed per page
```

```
# Custom viewset, inheriting from ModelViewSet
class ViewSet(viewsets.ModelViewSet):
    queryset = ExampleModel.objects.all() # Queryset, defining the database model instances to operate on
    serializer_class = Serializer # Serializer class, defining how data is serialized and deserialized
    pagination_class = Pagination # Pagination class, defining the pagination method
```

In the code, a Pagination class is added based on the original ViewSet. **Each class is 10, which means that one page displays ten rows of data.**

Filtering and Searching

Another concept is filtering and searching. In hardware, filtering waves removes unwanted frequency bands. Similarly, in software, filtering removes unwanted information, retaining only what is needed. When querying data, it is possible to search for specific content of interest. For example, to focus on data from the max30102 sensor, other parts can be filtered out, retaining only the relevant data for max30102.

```
from rest_framework import filters, viewsets

# Custom viewset, inheriting from ModelViewSet
class ViewSet(viewsets.ModelViewSet):
    queryset = Model.objects.all() # Queryset, defining the database model instances to operate on
    serializer_class = Serializer # Serializer class, defining how data is serialized and deserialized
    filter_backends = [filters.SearchFilter, filters.OrderingFilter] # Filter backends, defining the methods for searching and ordering
    search_fields = ['name'] # Search fields, defining the model fields that can be searched
    ordering_fields = ['date'] # Ordering fields, defining the model fields that can be ordered
```

The code is still within the viewset file because it pertains to the views. This is the visualization part, hence it belongs here. **The filter_backends imports two types of filters**, both of which are built-in functions and can be used directly. **SearchFilter is for searching, and OrderingFilter is for ordering. The search_fields is used to search based on names, while ordering_fields is used to sort based on dates.**

Functional summary: The main function of this code **is to create a view set with search and sorting functions. By defining the filter_backends attribute, the view set supports search and sorting functions:**

- Search function: Through SearchFilter, users can search on the fields defined in search_fields (here is name).
- Sorting function: Through OrderingFilter, users can sort on the fields defined in ordering_fields (here is date).

The view set ViewSet uses these filtering functions and defines the model Model for data query and the class Serializer for data serialization.